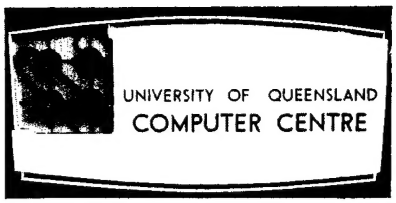


COMPUTER CENTRE BULLETIN

Vol. 4, No. 10
4 October 1971

Editor:
Mrs Sarah Barry

WORKSHOP ON 'PICTORIAL ORGANIZATION AND SHAPE'

A Workshop will be held at the C.S.I.R.O. Division of Computing Research, Canberra on the 29-30 November 1971. This date follows the Canberra Perception Symposium from 26-28 November. The aims of the Workshop are:

1. to discuss viewpoints on what constitutes 'pictorial organization' and 'shape'
2. to compare different algorithmic approaches for recovering pictorial organization and shape
3. to examine the results of psychological and physiological experiments which might be relevant to these approaches.

C.S.I.R.O. invites prospective attendees to reply by 5 October stating their background and interests. Intending speakers should also include a summary of their topic and be prepared to provide a written paper at the Workshop.

Reply directly to Dr J.F. O'Callaghan, C.S.I.R.O., Division of Computing Research, P.O. Box 109, CANBERRA CITY, A.C.T. 2601.

NEW EDITOR

A new version of Editor (version 1.1) became operative on Wednesday 25 August.

Most of the new features are invisible to the user but they result in improved file handling by the Editor.

Other new features are as follows:

(a) '*' is not suppressed after ↑O.

When control-O is typed the printout from the program is suppressed. Previously the '*' signalling that the Editor was ready to accept a new command was also suppressed and users had to assume that the Editor was ready.

(b) DISK ERROR message

The message 'DISK ERROR' has been expanded to explain the exact nature of the error.

(c) Lines split across disk blocks

Previously, if a line would not fit into the space left in a disk block the Editor would null fill the rest of the block and place the line in a new block. In this version, the Editor makes maximum use of file storage and will split a line across two blocks in order to fill a block completely with meaningful data.

NEW BATCH SYSTEM

The new batch processing system has now been implemented on the PDP-10. Full details of the new Batch were given in last month's Bulletin.

Two errors, detailed below, may occur with the new Batch system. These errors are being investigated and it is anticipated that they will shortly be fixed. In the meantime users are asked to check their output carefully and report any discrepancies to the Computer Centre.

(a) Under some circumstances, input records will not be read correctly. If the fields read terminate before the end of the card, but additional data is punched on the card up to column 78 then the error is likely to be experienced. It will manifest itself by the overprinting of adjacent lines of output or the absorption of the first column of a successive card, so that data fields are apparently shifted one place left.

(b) If a run terminates with a monitor detected error, then under some circumstances, the end of the message may be overwritten by a portion of some previous input record.

COMPUTER CENTRE MANUALS

(a) DDT-10

Chapter 3 of Technical Manual No.12 (PDP-10 Utility Programs) has now been completed. It describes the operation and use of DDT-10 (Dynamic Debugging Technique) in detail, and is available from the University Bookshop for $1.65.

(b) Systems User's Guide Revision

The first set of revision pages for the System User's Guide is now available. These may be obtained from the Computer Centre free of charge.

CORE STORAGE ALLOCATION FOR THE FORTRAN USER

Following the recent implementation of a new FORTRAN system that affected a number of users running very large programs, the Centre has received enquiries about the actual amount of core storage available for a user's FORTRAN program.

It is difficult to define a firm figure for the core area available for a user's program. Of the 24K overall limit, 4K is always required by the basic FORTRAN operating system. This handles input/output, format control and character conversion. The remaining 20K of core must contain the user's program, his data areas and all the other FORTRAN library facilities requested by his program. As well as the standard library routines of SQRT, SIN, COS, LOG, etc., various library routines are called for such operations as double precision or complex arithmetic, exponentiation, and for statements such as DEFINE FILE, ENCODE, END, etc. Thus, the amount of core required by FORTRAN is a variable quantity depending upon the particular facilities requested.

While the Centre will always endeavour to keep the maximum amount of core storage available to the user, future software releases from manufacturers may result in variations to the boundaries given above. It is suggested that users running large programs segment their programs into overlays or arrange for large data areas to be stored on disk.

FORTRAN VERSION 23

1. Warnings

Users should be aware of the following points when using FORTRAN.

(a) A format specifier in say a read, write, encode or decode statement can only refer to a format statement or an array which at the time of execution will contain a legitimate format string. If other than these 2 possibilities occur, the error will not be detected until an attempt is made to execute the statement. At present, the FORTRAN operating system attempts to output the character in the format string which is illegal (not output correctly at present) and also the format string which contained it. If the format specifier referred to say the statement ending a DO loop, then the execution package will attempt to output locations of the user's program as a character string, with generally unuseful results. A change is in the process of being implemented that will list properly the character discovered as being in error, and subsequently a more meaningful message will be output if the string pointed at is found not to be a legitimate character string.

(b) Subprogram names may not be used as dummy arguments or appear in any non-executable statement in a program other than as a scalar variable in a type statement. It must appear as a scalar variable and be assigned a value during execution of the subprogram which is the function value.

e.g. SUBROUTINE A(A) is incorrect.

(c) A number of FORTRAN compiler diagnostics are either not detected or not flagged correctly

e.g. I-2 ARRAY NAME ALREADY IN USE is flagged as S-1 SYNTAX

M-12 NON INTEGER PARAMETER is flagged as S-10 ILLEGAL CHARACTER

When the compiler detects an error it records the error and the continuation card and column at which the error occurred. The compiler then returns to the statement recognition scan routine. Depending on the nature of the initial error, it may be that other error situations will be induced. The compiler assumes the last error is the correct one and ignores previous ones. In practice, this means that the diagnostic message may not be very meaningful, although it does indicate an error of some sort exists in the statement.

(d) When using free field input users should be careful not to use a mixture of delimiter characters between adjacent fields. Blanks or any non-standard character can be used as field delimiters, but combinations of these will result in input variables being set to zero as the input routines treat a change in delimiter character as a null field.

example:

The following program:

```
 2    WRITE (6,1)
 1    FORMAT (' 2 REAL & 2 INTS'/)
      READ (5,5) A,B,J,K
 5    FORMAT (2F,2I)
      WRITE (6,1Ø) A,B,J,K
1Ø    FORMAT (' ',2F,2I)
      GO TO 2
      END
```

provides the following results:

```
 2 REAL + 2 INTS
1.5Δ2.6Δ4Δ78
      1.5ØØØØØØ        2.6ØØØØØØ        4        78

 2 REAL + 2 INTS
23.,5.6Δ7<tab>8Ø
     23.ØØØØØØØ        5.6ØØØØØØ        7        8Ø

 2 REAL + 2 INTS
12.5Δ,Δ4.59ΔΔΔ3,2
     12.5ØØØØØØ        Ø.ØØØØØØØ        4        59

 2 REAL + 2 INTS
↑C
```

(e) Users should beware of specifying constants as the arguments of a call to a routine when that routine involves the exchange of the values of its arguments.

example:

Main program -

```
                .
                .
                .
                .
                X = SOM (2.Ø,3.Ø,Z)
                .
                .
                .
                .
```

Function -

```
                FUNCTION SOM (A,B,M)
                IMPLICIT - - -
                IF (A.LE.B) GO TO 1Ø
                C = A
                A = B
                B = C
      1Ø        .
                .
                .
```

The result of the call to SOM from the main program will exchange the actual values of the constants 2.Ø and 3.Ø. And thereafter 2.Ø will have a 'value' of 3 and 3.Ø will have a 'value' of 2.

These errors have been reported to Digital for correction.

(a) Users should not use repeats with slashes in FORMAT statements.

e.g. 5F4.Ø will work correctly

but (2(/), 5F4.Ø) will cause values to be lost, in this example every sixth output value is lost.

(b) In the following example the output list contained 3 implied DO loops with an overall fourth DO loop. The outer two have their first index as a variable (J), but the middle one has a constant index (1). Wrong code is produced for the third implied DO loop.

```
IMPLICIT INTEGER (A-Z)
DIMENSION THEM (9,3ØØ)
WRITE (6,3Ø5) FMT, (THEM(J,I),J=2,7),THEM(1,I),(THEM(J,I),J=8,
1ITMSIZ), I=INDEX,MOST,LINES)
FORMAT (A1,2ØX,6A5,2X,A1,1X,2A5,2ØX,6A5,2X,A1,1X,2A5)
END
```

The cure is to rearrange the data so that the same first index is used for the three internal implied DO loops.

3. These errors have been corrected, but the patches have not yet been implemented into the system.

(a) Users are warned not to use mixed mode expressions that involve subexpressions of integer, real and double precision type. If integer expressions are avoided, results are satisfactory. However, if integer expressions are involved, the compiler in some circumstances fails to take note of the 'type' of the variables when converting from integer to real to double.

(b) Statements involving logical IFs followed by subroutine CALLS result in an illegal UUO.

e.g. `IF (IC.GT.Ø) CALL PUTOUT (')')`

results in two arguments being generated and the program will crash with an illegal UUO.

(c) Recursive statements are not flagged as illegal by the FORTRAN compiler. However the code generated by the statement will cause the program to enter a loop from which it cannot exit.

e.g. `FUNC (I,J,K)=I+J+FUNC(I,J,K)`

(d) The compiler generates incorrect subroutine exit code when some of the dummy arguments are double precision or complex arrays.

(e) The compiler does not always handle complex arithmetic correctly, e.g. in the case of division of a complex number the imaginary part is not divided.

(f) The file separator character is not detected by the operating system when reading under any format other than type A.

(g) OFILE sometimes produces the wrong extension to a data file.

e.g. `FOR1Ø/Z$C`

(h) Inputting a Hollerith string into a format statement containing single quotes has an error in that the next character to be input is ignored,

e.g. input under ('ΔΔΔΔΔ',I4)

ABCDE1234

is then output as

ABCDE234Ø

Users should note also that inputting a single quote character into a format statement or array will result in the character translated on input to the double quote character. This may seem anomalous but is a reasonable action if single quotes are not to be illegal in this situation.

Programmers are advised to use H type rather than single quotes for inputting Hollerith strings to avoid both these problems,

e.g. ('ΔΔΔΔΔ',I4)

should be replaced by

(5HΔΔΔΔΔ,I4)

if this is to be used as an input format statement. The same remarks apply to formats in arrays.

example:

The following program

```
    1    PRINT 1Ø
    1Ø   FORMAT (' INPUT UNDER (''SSSSS'',I4)'/)
         READ 11,INT
    11   FORMAT('SSSSS',I4)
         PRINT11,INT
         GO TO 1
         END
```

produces the following results:

```
INPUT UNDER ('SSSSS',I4)
.....1234
....234Ø
INPUT UNDER ('SSSSS',I4)
..'..12345
."..2345
INPUT UNDER ('SSSSS',I4)
'''''12345
"""""2345
```

BASIC VERSION 15

An illegal statement, such as

```
1Ø    A=1, B=Ø
```

is not detected and is not flagged with a diagnostic message. Instead some kind of execution is attempted, usually giving erroneous results. In PDP-10 BASIC it is incorrect to write two LET statements on one line, but the compiler has failed to recognize the error and the program results in an execution error. Users should check the syntax of their programs as well in these cases if they feel that there is no error in their program logic.

MACRO VERSION 43

Macro expansion fails when an IRP is contained inside a REPEAT within a Macro.

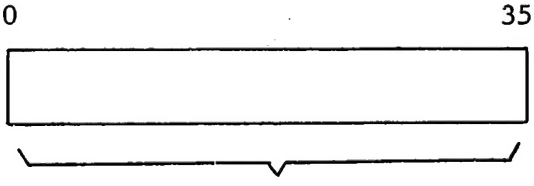
PLOTTING

The problem reported in WN-43 concerning plotter coordinates close to the plotting boundary appears to have been solved. The Computer Centre would appreciate any users still having trouble with this problem to contact the Administrative Officer.

FUNCTION SUBPROGRAM FOR BYTE MANIPULATION

A real function subprogram called BYTE and an integer function subprogram called IBYTE are now available in the FORTRAN library. BYTE and IBYTE allow the user to manipulate bytes in a FORTRAN program.

A byte on the PDP-10 is a collection of consecutive bits, from 1 to 36 bits long. Bits are numbered from 0 to 35.



This function allows the programmer to 'lift' a byte of any length from any position in a source word. The value of the result of the function is the value of the destination word with the specified byte from the source word inserted in the required position. The values of all the arguments are unchanged.

The call to the function is as follows:

$$\begin{Bmatrix} \text{VALUE} \\ \text{IVALUE} \end{Bmatrix} = \begin{Bmatrix} \text{BYTE} \\ \text{IBYTE} \end{Bmatrix} \text{(SOURCE, LENGTH, IS, DEST, ID)}$$

SOURCE is the source word containing the byte

LENGTH is the length of the byte in bits

IS is the leftmost bit of the byte in SOURCE

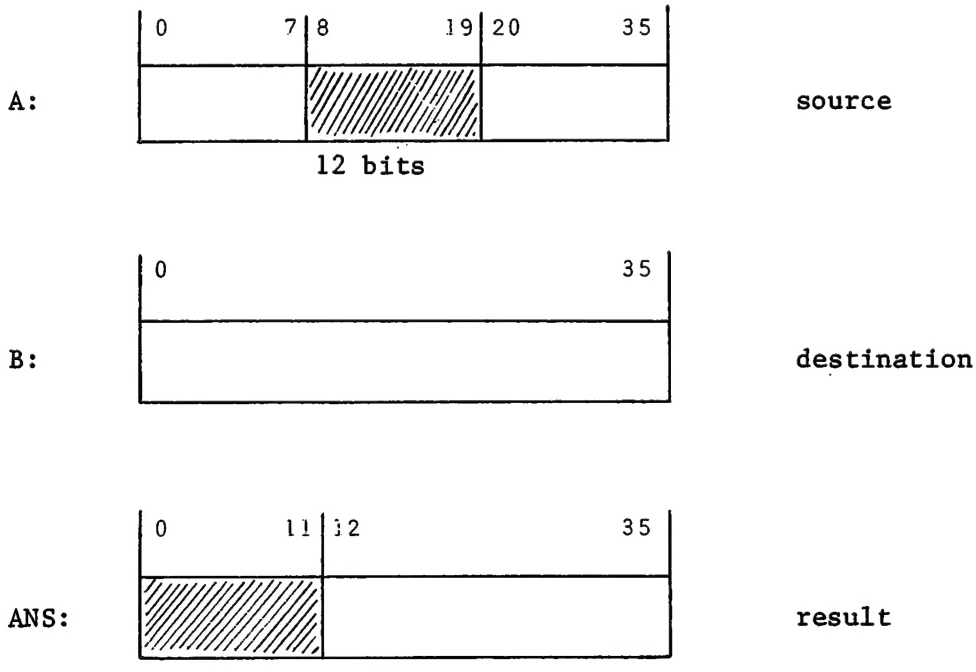
$\begin{Bmatrix} \text{VALUE} \\ \text{IVALUE} \end{Bmatrix}$ is the result of the call. It contains the value of the word DEST with the byte inserted. ID is the leftmost bit where the byte is placed.

LENGTH, IS and ID are integers.

examples:

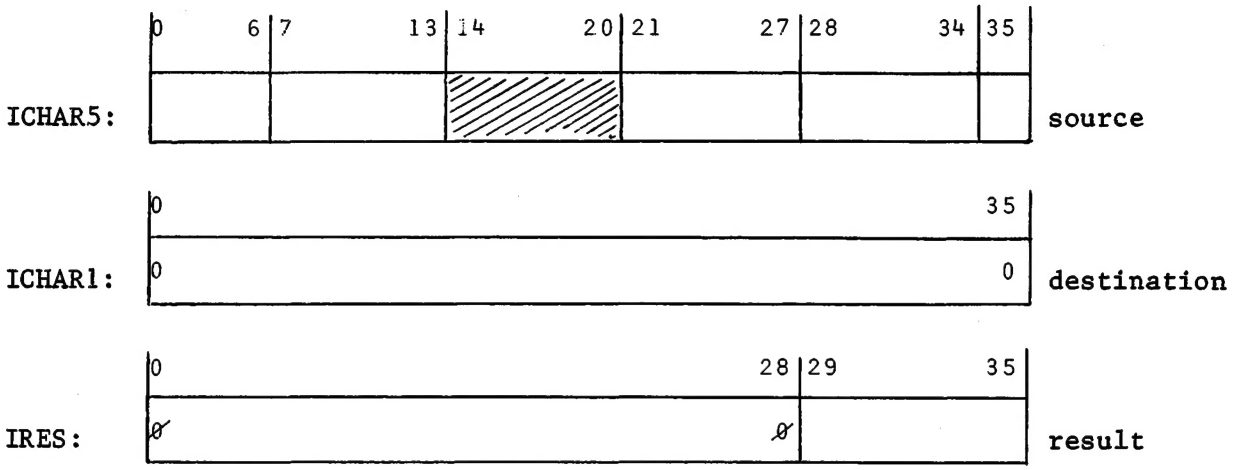
(i) If we wished to obtain a result with the value of B containing a byte of 12 bits obtained from A, the call would be:

ANS = BYTE (A, 12, 8, B, Ø)



(ii) If we wished to obtain the third character in a word and deposit it right justified with zero filling in IRES, the call would be

IRES = IBYTE (ICHAR5, 7, 14, Ø, 29)



The following non-fatal error messages could occur.

(i) LENGTH < 0 or LENGTH > 36
returns the result as DEST and gives the message
BYTE ARGUMENT OUT OF RANGE

(ii) $\{^{IS}_{ID}\}$ < 0 or $\{^{IS}_{ID}\}$ > 35

returns the result as DEST and gives the message
BYTE ARGUMENT OUT OF RANGE

(iii) LENGTH + ID > 35
returns the result as DEST and gives the message
BYTE CROSSES WORD BOUNDARY